The Challenges of X86 Hardware Virtualization

The Challenges of X86 Hardware Virtualization

- X86 operating systems are designed to run directly on the bare-metal hardware, so they naturally assume they fully **‘own’** the computer hardware
- X86 architecture offers four levels of privilege known as Ring 0, 1, 2 and 3 to operating systems and applications to manage access to the computer hardware

The Challenges of X86 Hardware Virtualization Conti..

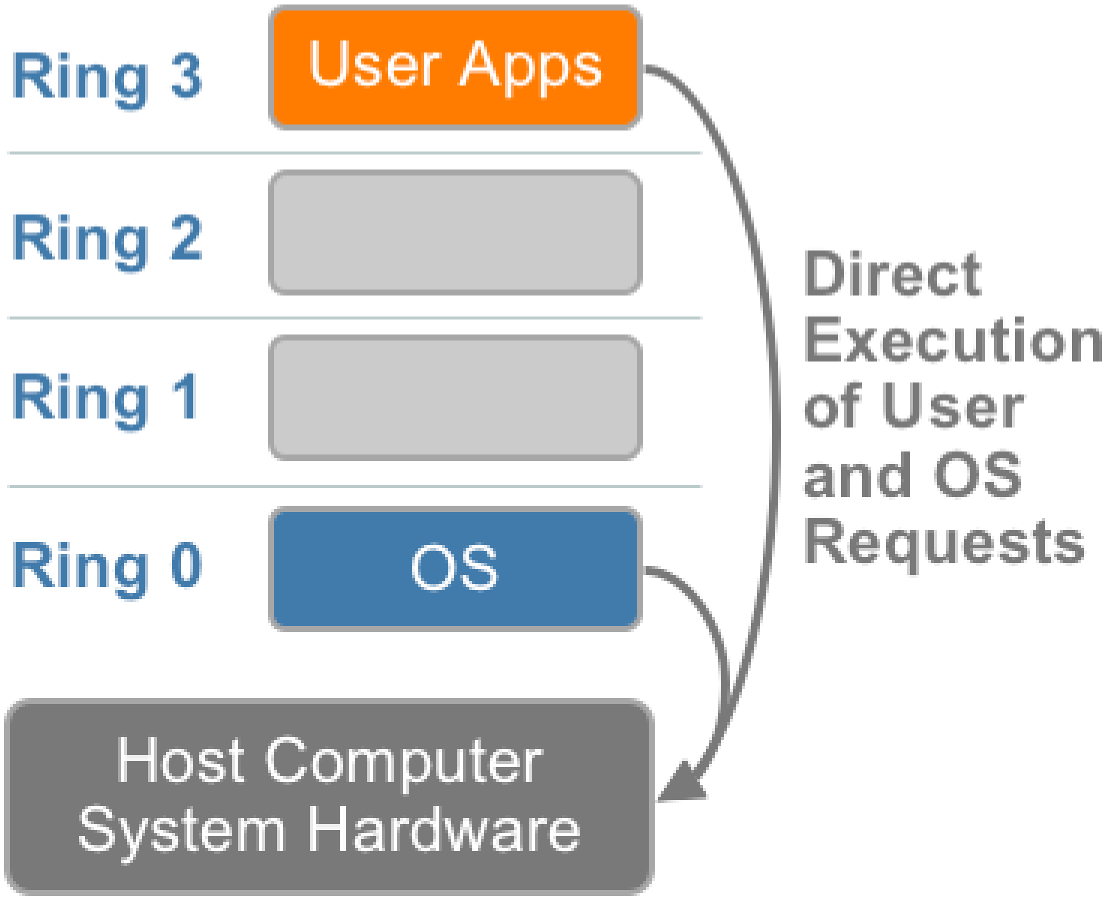


Figure Courtesy: Understanding Full Virtualization, Paravirtualization, and Hardware Assist, VMware

The Challenges of X86 Hardware Virtualization Conti..

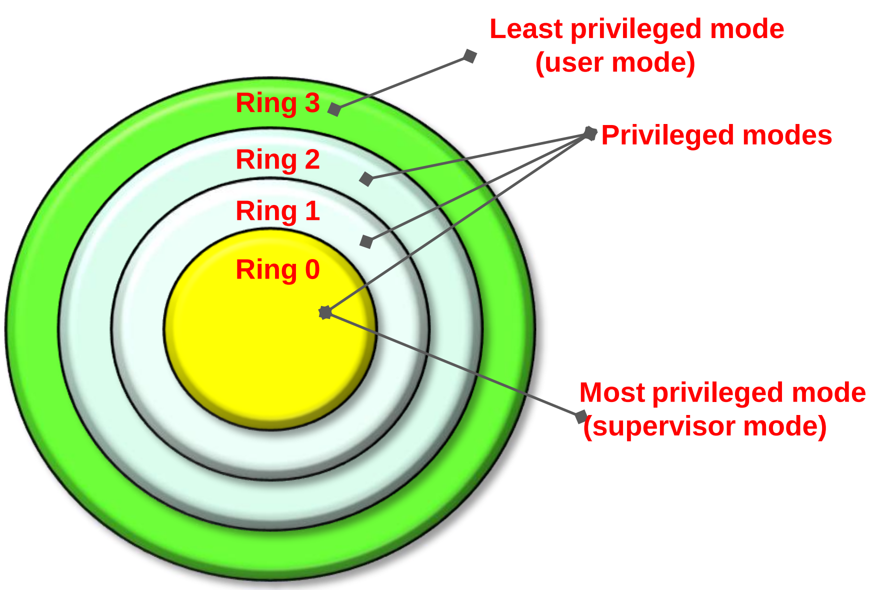
- Virtualizing the X86 architecture requires placing a virtualization layer under the operating system to create and manage the virtual machines that deliver shared resources.
- *Some sensitive instructions can't effectively be virtualized as they have different semantics when they are not executed in Ring 0.*

The Challenges of X86 Hardware Virtualization Conti..

- *The difficulty in **trapping** and **translating** these sensitive and privileged instruction requests at runtime was the challenge that originally made X86 architecture virtualization look impossible.*

Security Rings and Privileged Modes

- Ring 0 is used by the kernel of the OS and rings 1 and 2 are used by the OS level services and Ring 3 is used by the user.
- Recent systems support only two levels with Ring 0 for the supervisor mode and Ring 3 for user mode



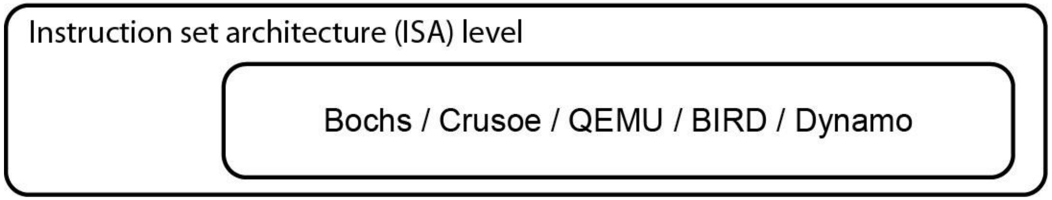
Supervisor mode

- If code is running in *supervisor mode* all the instructions (privileged and non-privileged) can be executed without any restriction.
- This mode is also called *master mode*, or *kernel mode* and it is generally used by the OS (or the hypervisor) to perform sensitive operations on hardware level resources.

User mode

- If code running in user mode invokes the privileged instructions, hardware interrupts occur and trap the potentially harmful execution of the instruction.

Five Abstraction Levels



Five Abstraction Levels

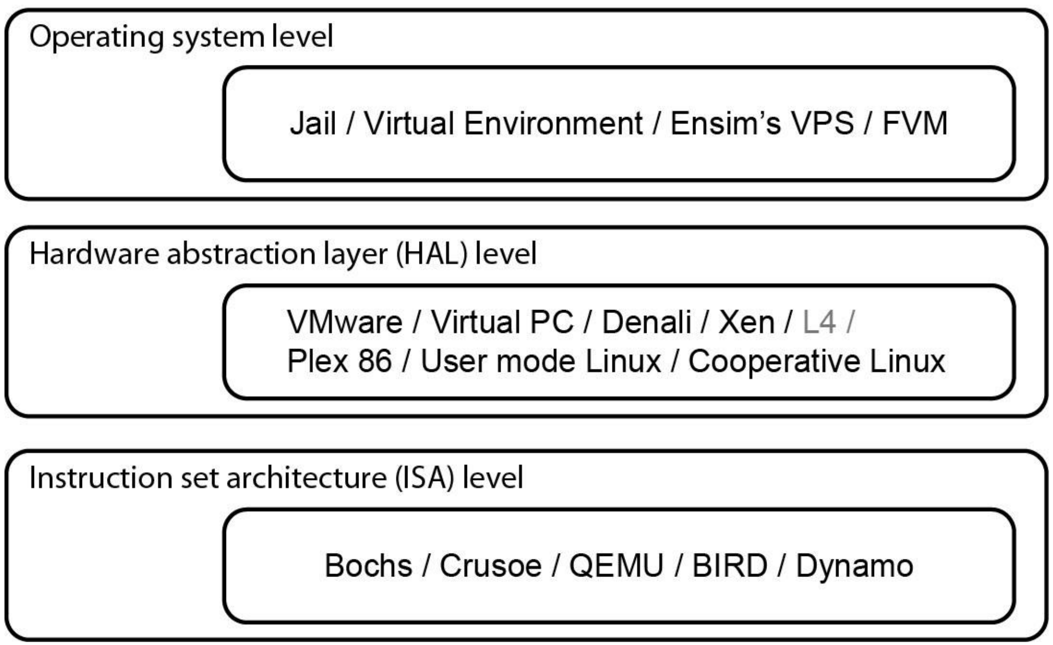
Hardware abstraction layer (HAL) level

VMware / Virtual PC / Denali / Xen / L4 /
Plex 86 / User mode Linux / Cooperative Linux

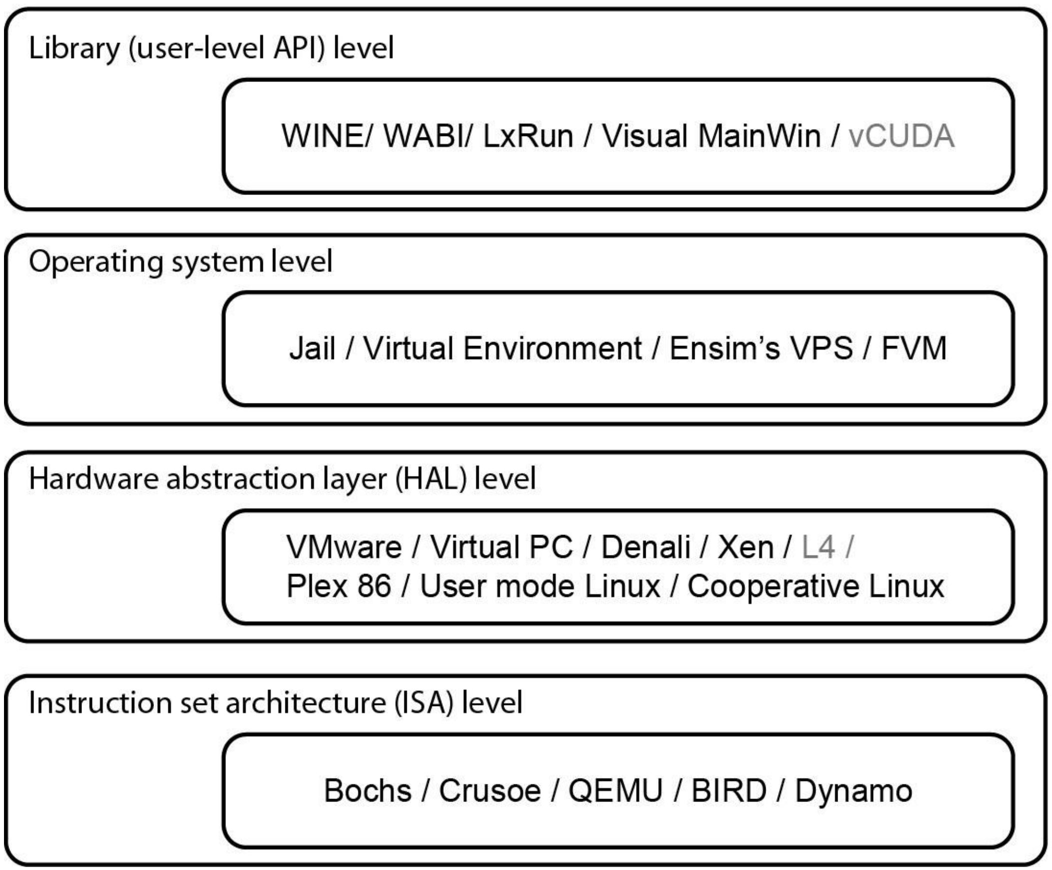
Instruction set architecture (ISA) level

Bochs / Crusoe / QEMU / BIRD / Dynamo

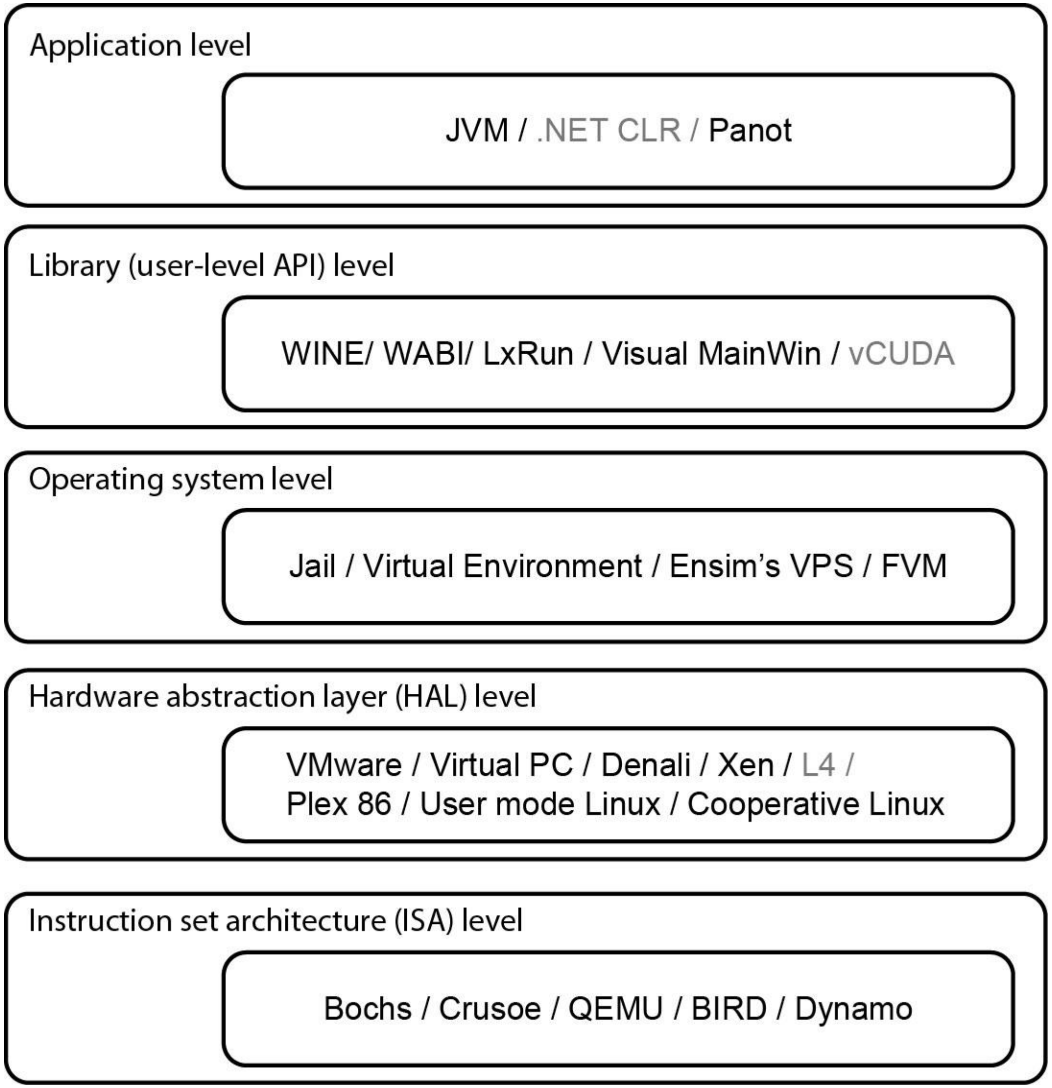
Five Abstraction Levels



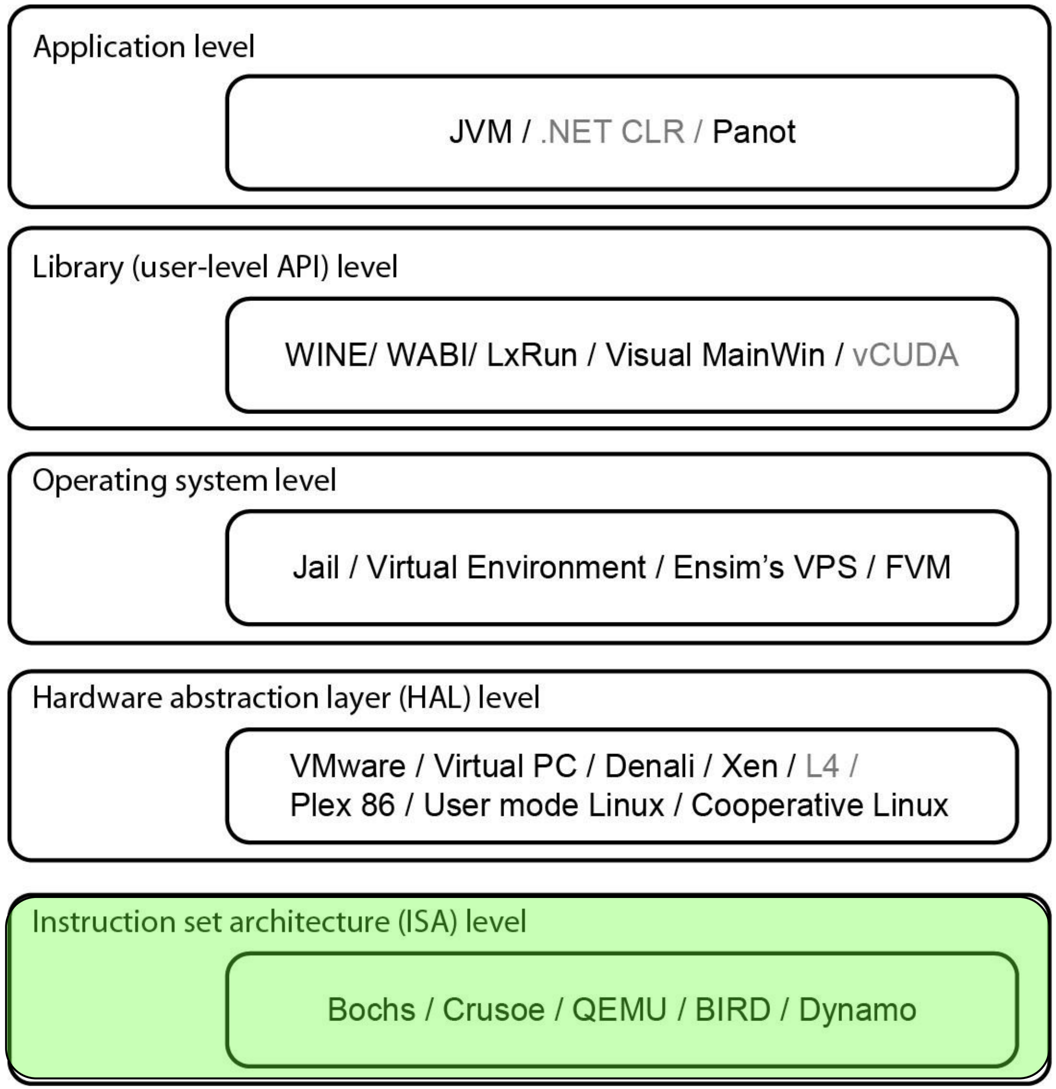
Five Abstraction Levels



Five Abstraction Levels



Five Abstraction Levels



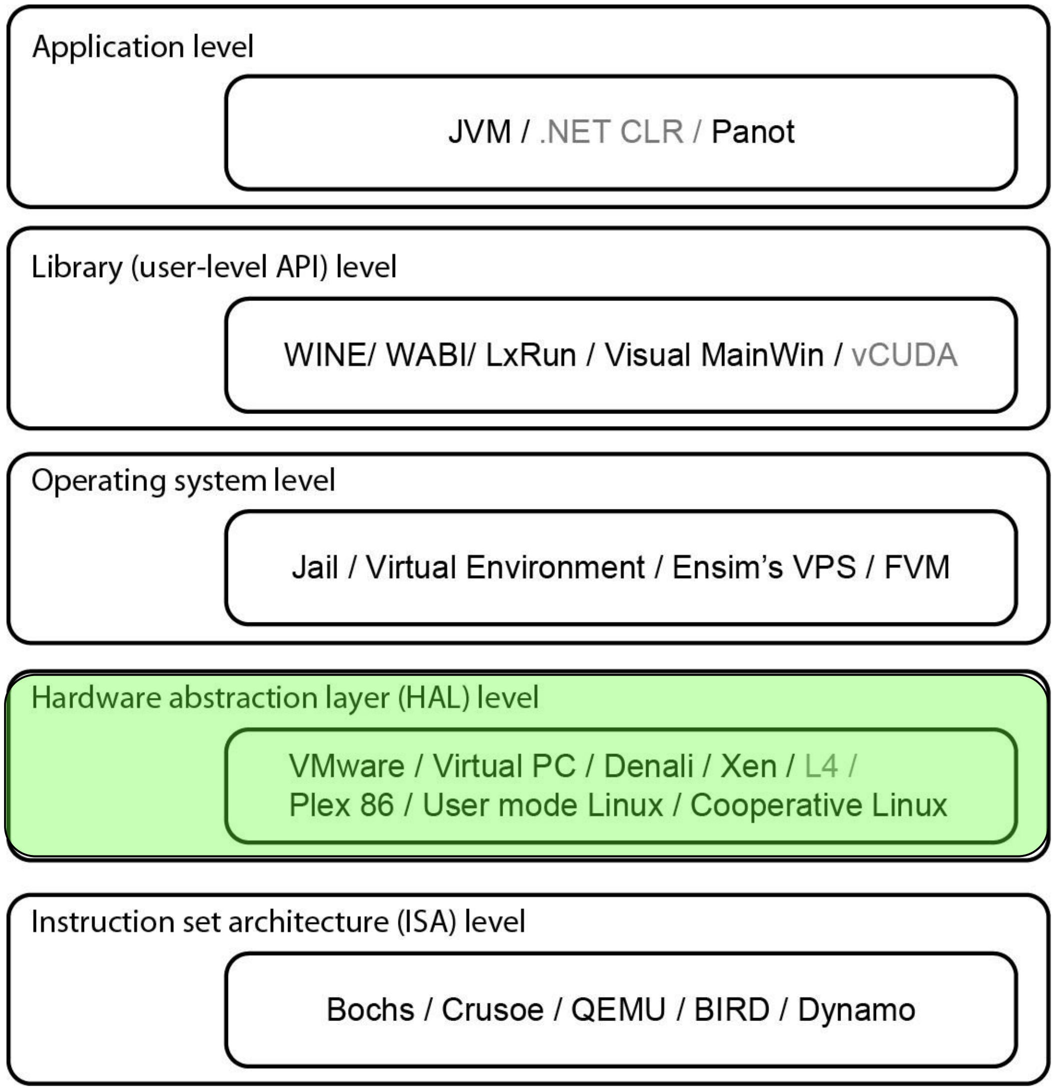
Instruction Set Architecture Level

- Virtualization is performed by emulating a given ISA by the ISA of the host machine.
  - MIPS binary code can run on an X86-based host machine
- The basic emulation method is through *code interpretation*
  - interprets the source instructions to target instructions one by one

Instruction Set Architecture Level Conti..

- For better performance, *dynamic binary translation* is used. This approach translates basic blocks of dynamic source instructions to target instructions
- Instruction set emulation requires binary translation and optimization.
- V-ISA thus requires adding a processor-specific software translation layer to the compiler.

Five Abstraction Levels



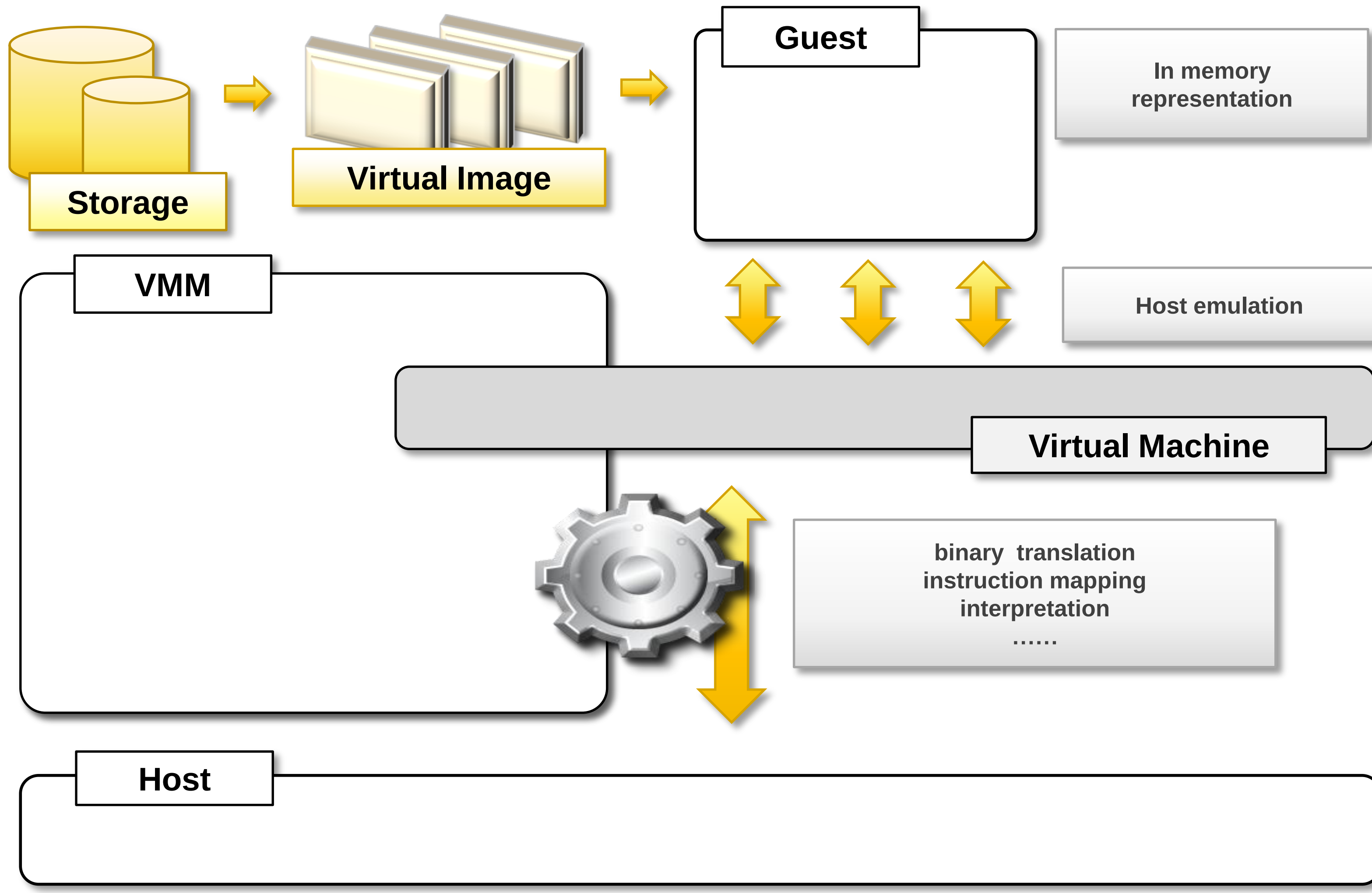
Hardware Level Virtualization

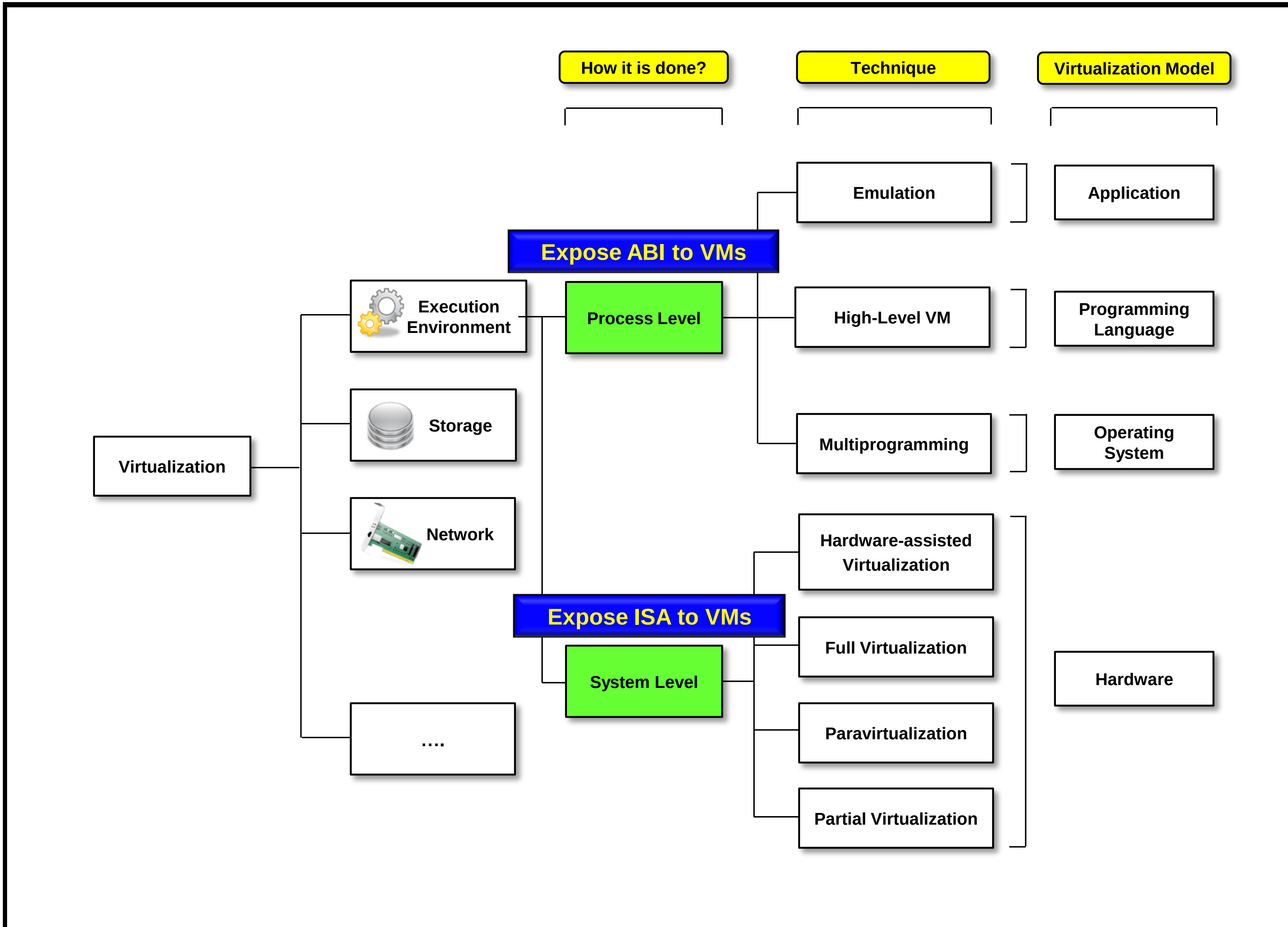
- Virtualization technique that provides an abstract execution environment in terms of computer hardware on top of which a guest operating system can run.
- In this model, the guest is represented by the *OS*, the host by the *physical computer hardware*, the virtual machine by its *emulation*, and Virtual Machine Manager by the *hypervisor*

Hardware Level Virtualization Conti…

- Hardware level virtualization is also called *system virtualization*, since it provides ISA to VMs, which is the representation of the hardware interface of a system.
- This is to differentiate from *process virtual machines*, which expose ABI to VMs.

Hardware Virtualization Reference Model





How it is done?
Technique
Virtualization Model
Emulation
Application
Expose ABI to VMs
Execution Environment
Process Level
High-Level VM
Programming Language
Storage
Multiprogramming
Operating System
Virtualization
Network
Hardware-assisted Virtualization
Expose ISA to VMs
Full Virtualization
System Level
Hardware
Paravirtualization
....
Partial Virtualization

Hypervisors

- A fundamental element of hardware virtualization is the hypervisor, or Virtual Machine Manager (VMM).
- It recreates a hardware environment, where guest operating systems are installed.

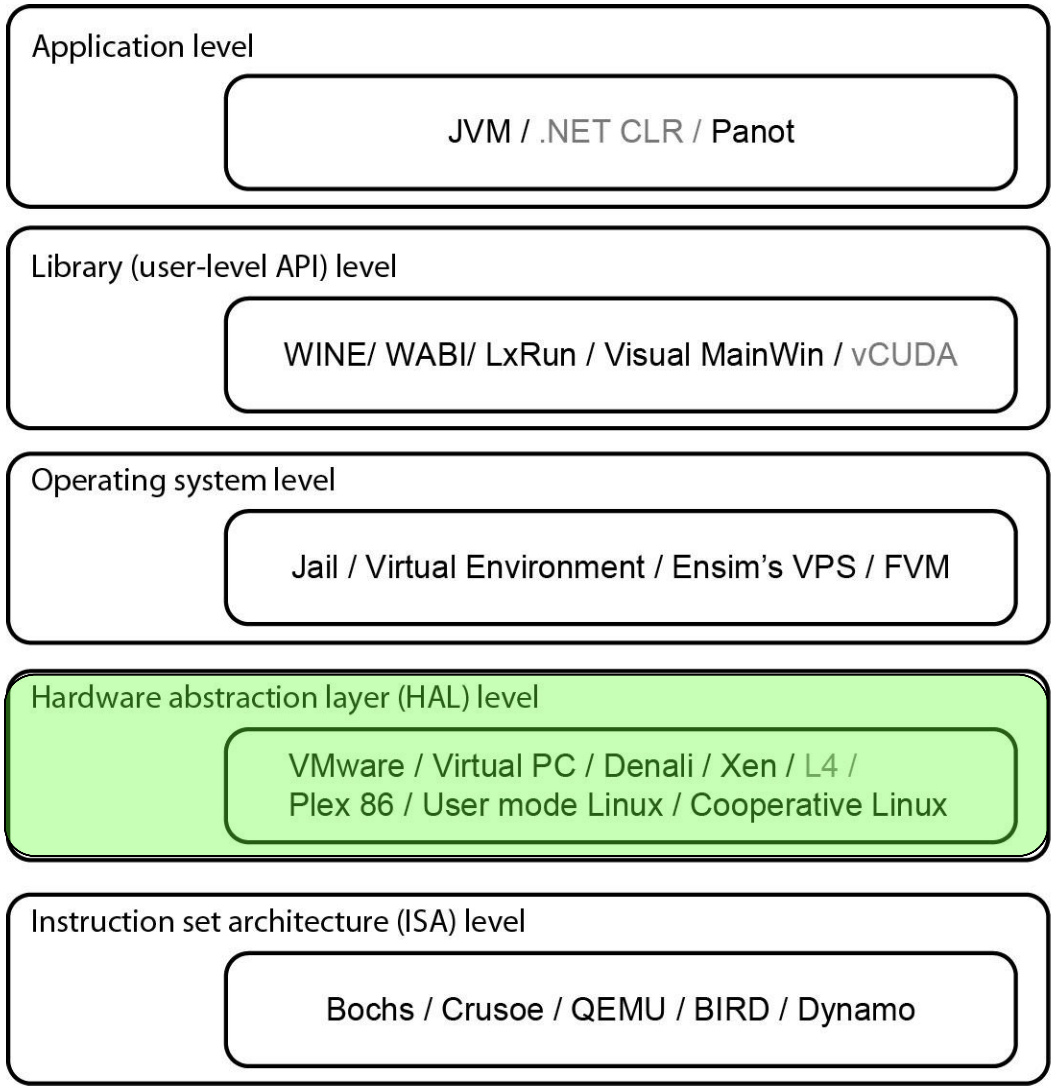
VMM Design Requirements

1. The VMM is **responsible** for allocating hardware resources for programs;
2. it is **not possible** for a program to access any resource not explicitly allocated to it; and
3. it is **possible** under certain circumstances for a VMM to regain control of resources already allocated.

- Not all processors satisfy these requirements for a VMM.

- There are two major types of hypervisors:
  - *Type I* and
  - *Type II*.

Five Abstraction Levels



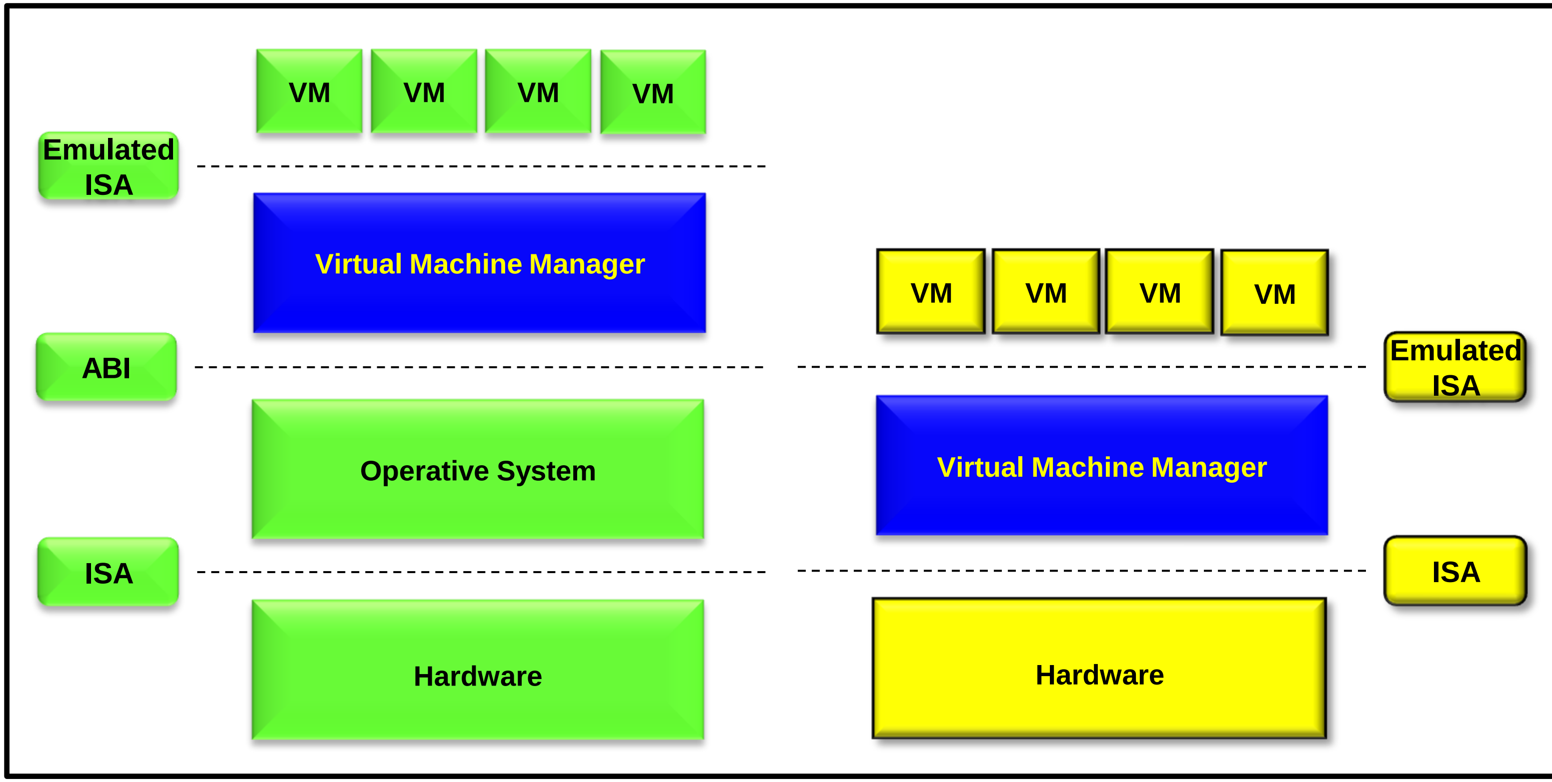
Type I Hypervisor

- *Type I* hypervisors run directly on top of the hardware. Therefore, they take the place of the operating systems.
- Interact directly with the ISA interface exposed by the underlying hardware, and *emulate ISA interface* in order to allow the management of the *guest OS*.
- This type of hypervisors is also called *native virtual machine*, since it run natively on hardware.

Type II Hypervisor

- *Type II* hypervisors require the support of an OS to provide virtualization services.
- *Type II hypervisors* are programs managed by the OS, that interacts with *OS* through the *ABI* and emulate the ISA of virtual hardware for the *guest OS*.
- This type of hypervisors is also called *hosted virtual machine*, since it is hosted within an operating system.

Hosted (left) and Native (right) VM

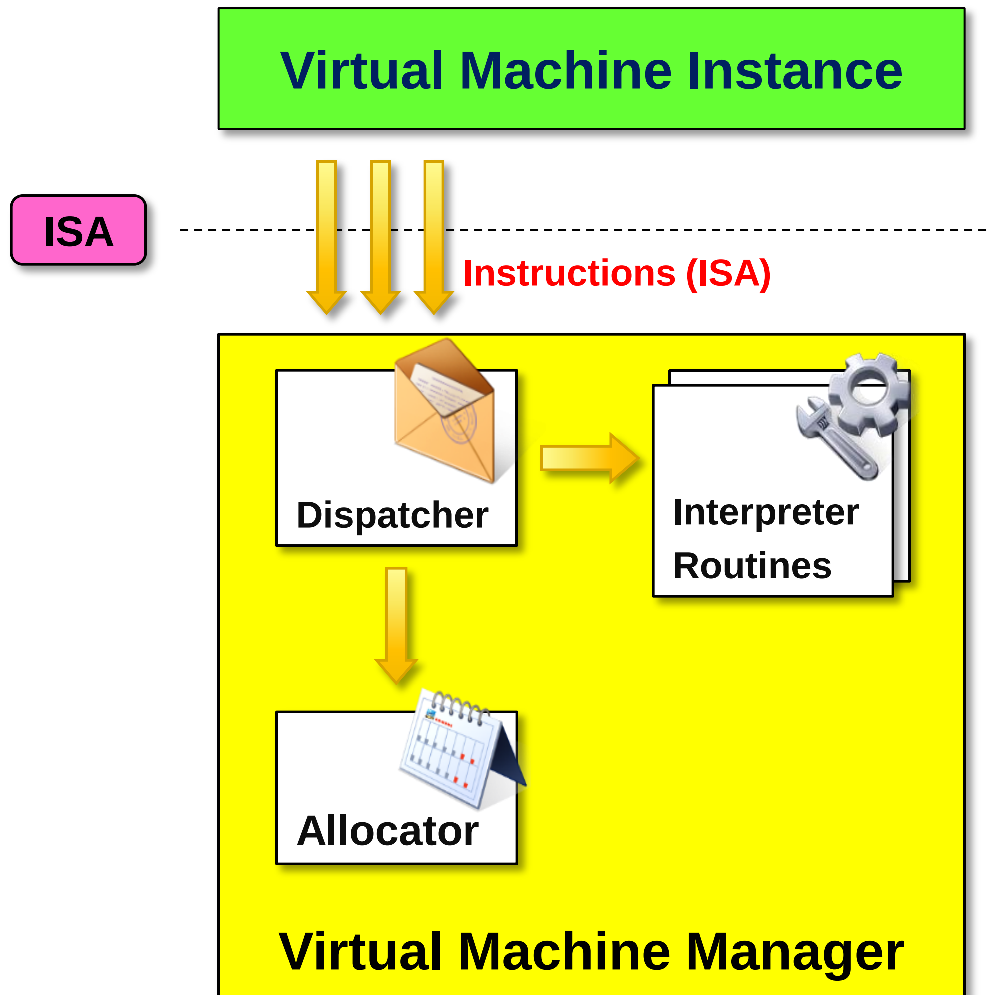


Hosted (left) and Native (right) VM

VMware Workstation	VMware ESX
KVM	Xen
Virtual PC & Virtual Server	Hyper-V

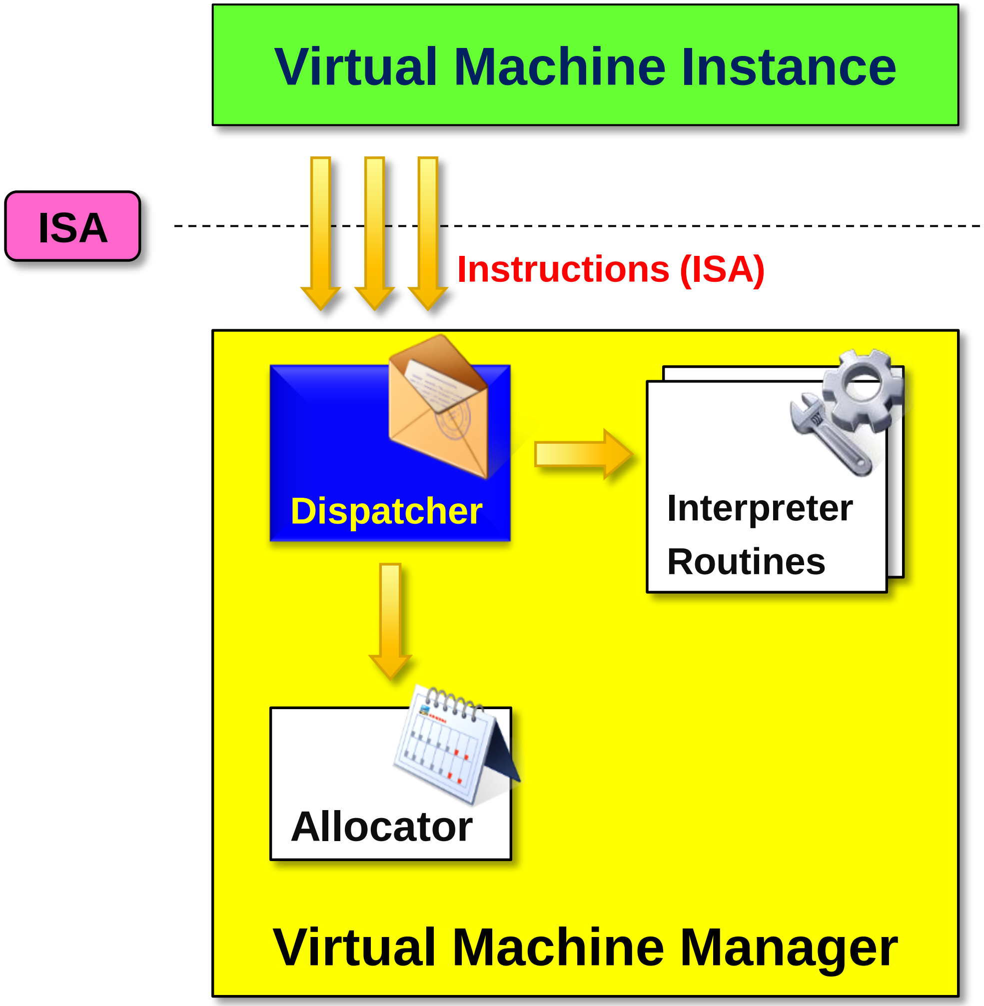
Internal Organization of VMM

- *dispatcher*, *allocator*, and *interpreter are* three main modules that coordinate activity in order to emulate the underlying hardware:



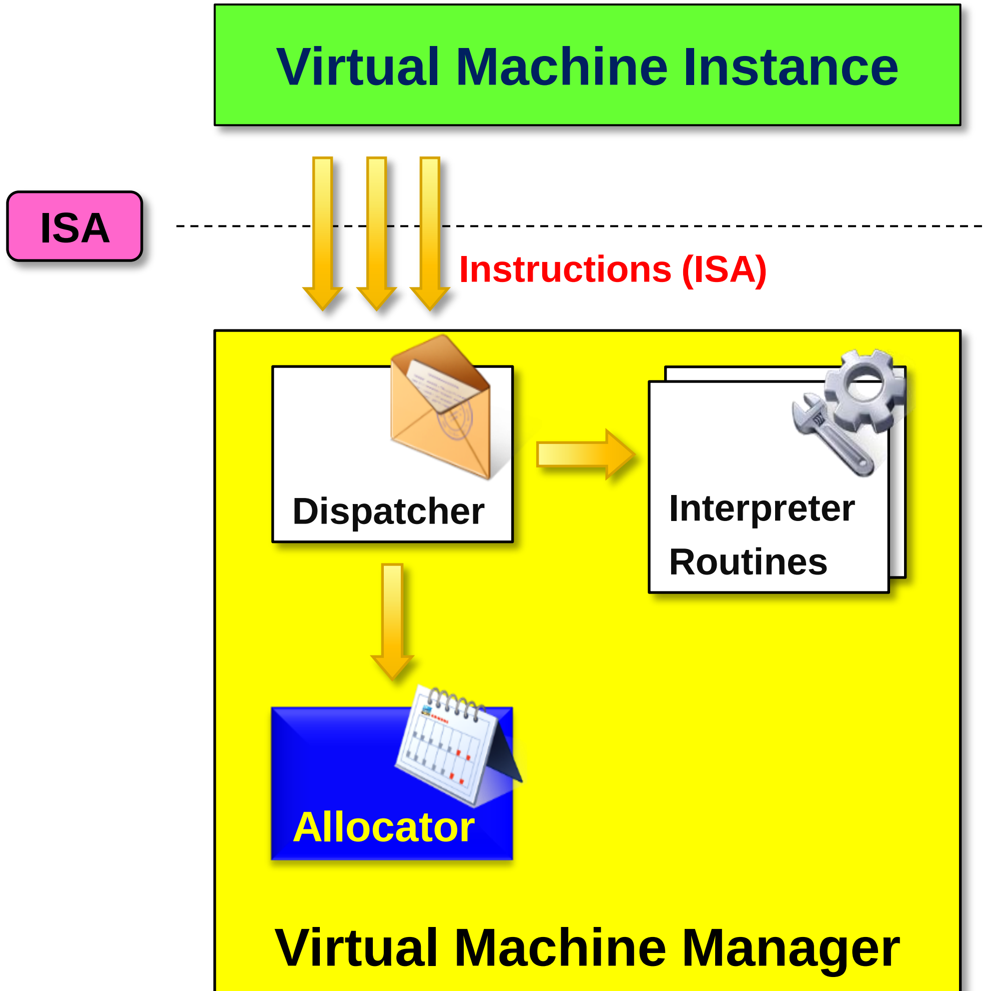
VMM Internal

- Dispatcher: entry point of the monitor and reroutes the instructions issued by the virtual machine instance to one of the two other modules



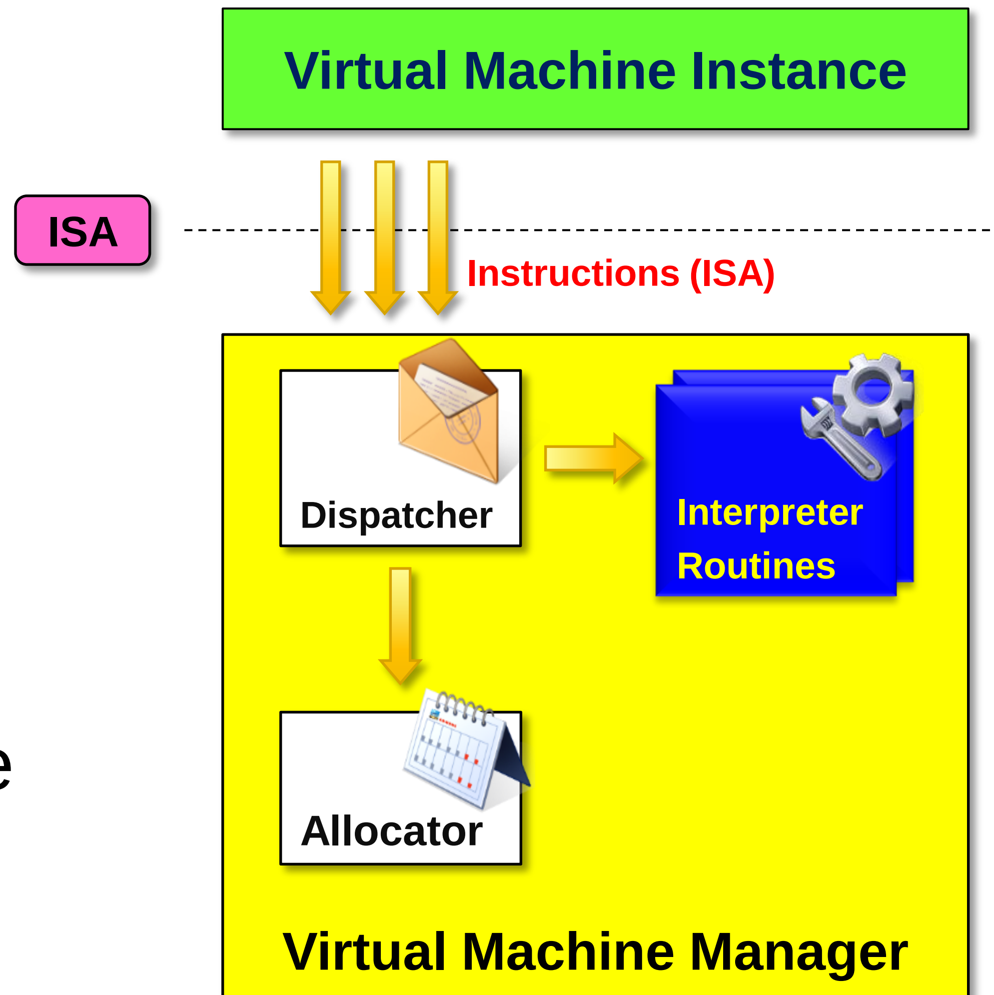
VMM Internal

- Allocator: is responsible for deciding the system resources to be provided to the VM



VMM Internal

- Interpreter: it consists of interpreter routines. These are executed whenever a VM executes a **privileged instruction**: a trap is triggered and the corresponding routine is executed.



VM Architecture

- The hypervisor provides hypercalls*** for the *guest OSes* and *applications*. Depending on the functionality, a hypervisor can assume a micro-kernel hypervisor architecture Or it can
- assume a monolithic hypervisor architecture for server virtualization

****“A hypercall is a software trap from a domain (domain is one of the virtual machines that run on the system) to the hypervisor, just as a syscall is a software trap from an application to the kernel”*

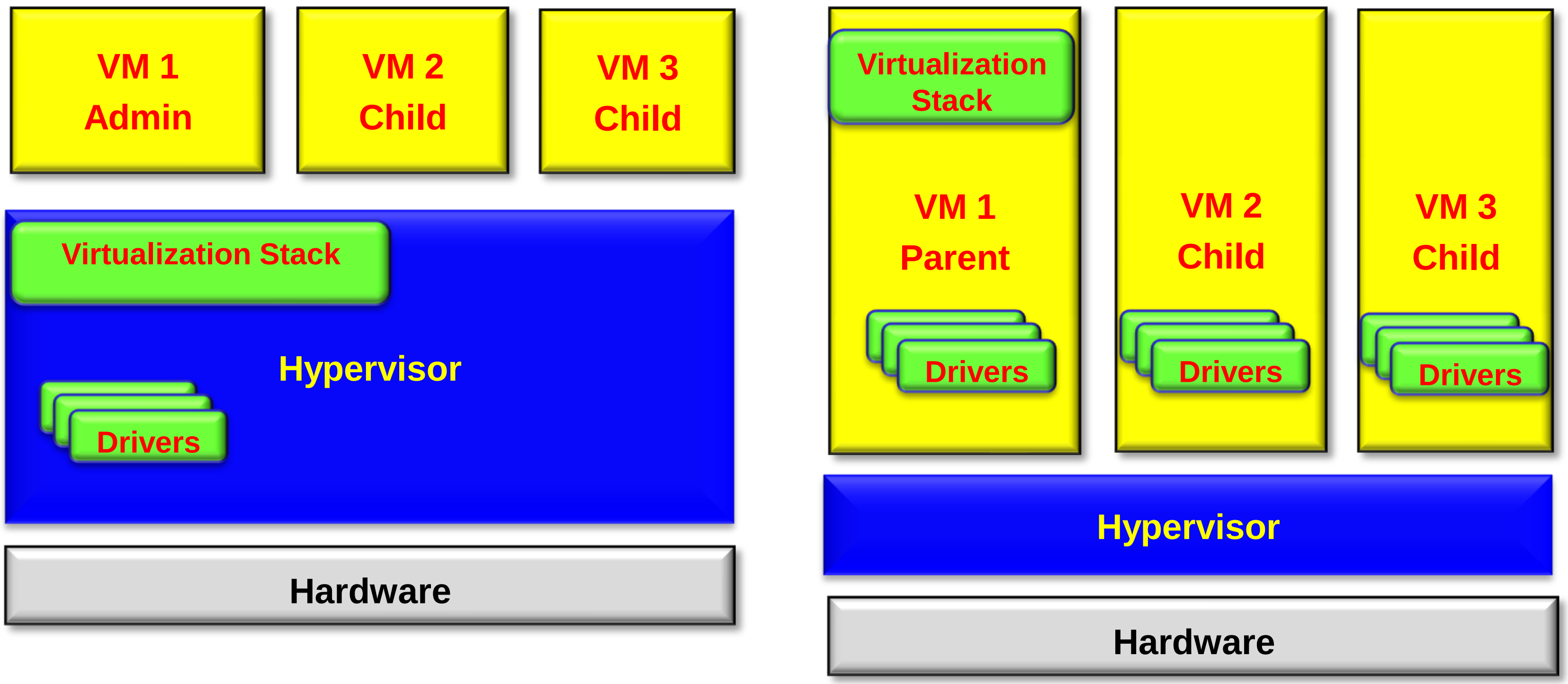
Continue..

- A micro-kernel hypervisor includes only the basic and unchanging functions ex.
    - physical memory management and
    - processor scheduling
- The *device drivers* and *other changeable components* are outside the hypervisor.

Continue…

- A monolithic hypervisor implements all the aforementioned functions, including those of the device drivers.
- Therefore, the size of the hypervisor code of a micro-kernel hypervisor is smaller than that of a monolithic hypervisor

Monolithic Vs Microkernel Hypervisor



- More simple than a modern kernel, but still complex
- Implements a driver model
- Third party vulnerability of drivers

- Simple partitioning functionality
- Increase reliability and minimizes Trusted Computing Base (TCB)
- No third-party code
- Drivers run within guests

V-Alternatives for X86 architecture

- **Three** alternative techniques exist for handling sensitive and privileged instructions to virtualize the CPU on the X86 architecture:
  - *Full virtualization using binary translation*
  - *OS assisted virtualization* or *paravirtualization*
  - *Hardware assisted virtualization (first generation)*

Binary Translation of Guest OS Requests Using a VMM

- This system puts the VMM at Ring 0 and the guest OS at Ring 1.
- The VMM scans the instruction stream and identifies the privileged, control and behavior-sensitive instructions.
- When these instructions are identified, they are trapped into the VMM, then **VMM emulates the behavior of these instructions**.
- The method used in this emulation is called *binary translation.*

Binary Translation of Guest OS Requests Using a VMM Conti…

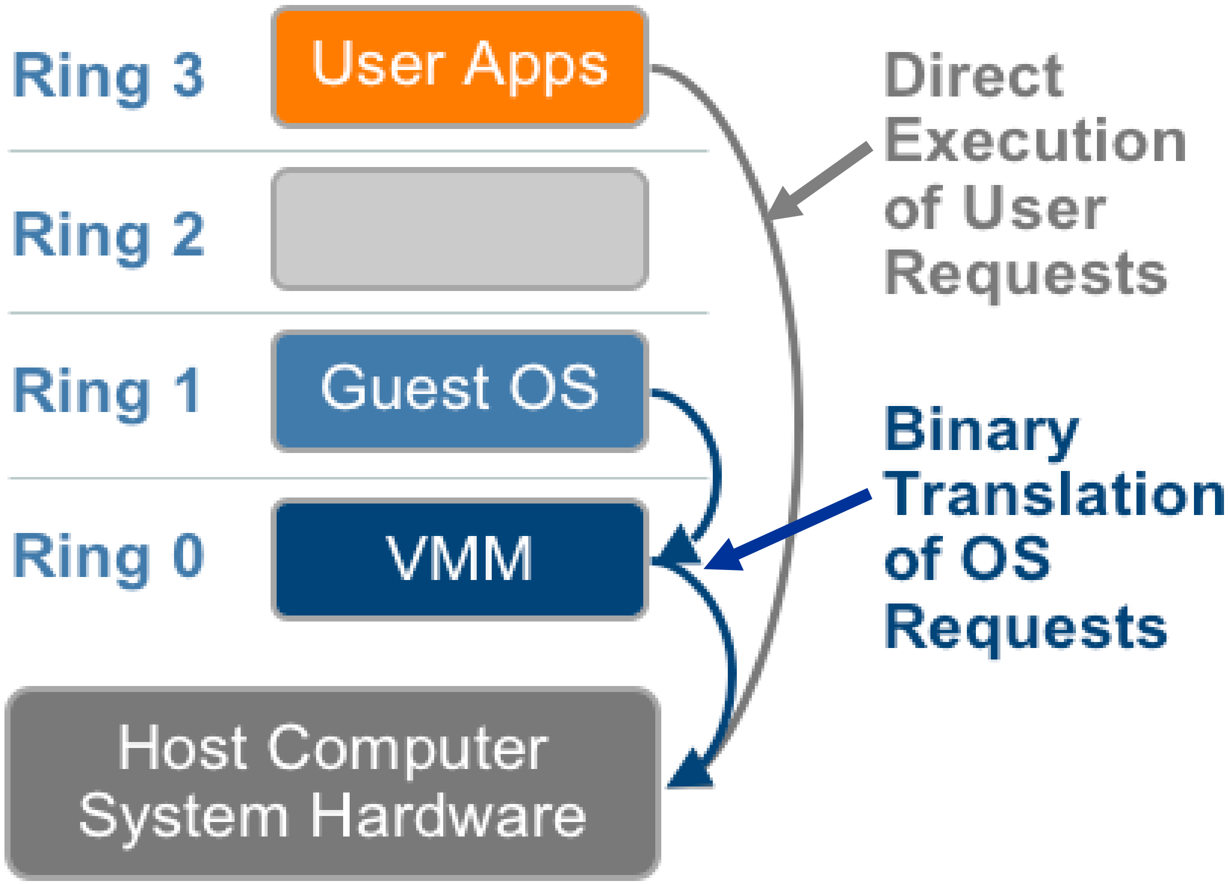


Figure Courtesy: Understanding Full Virtualization, Paravirtualization, and Hardware Assist, VMware

Binary Translation of Guest OS Requests Using a VMM Conti…

- Therefore, full virtualization combines binary translation and direct execution.
- The guest OS is completely decoupled from the underlying hardware.
- Consequently, the guest OS is unaware that it is being virtualized.
- The method is known a *Full Virtualization with binary translation*.

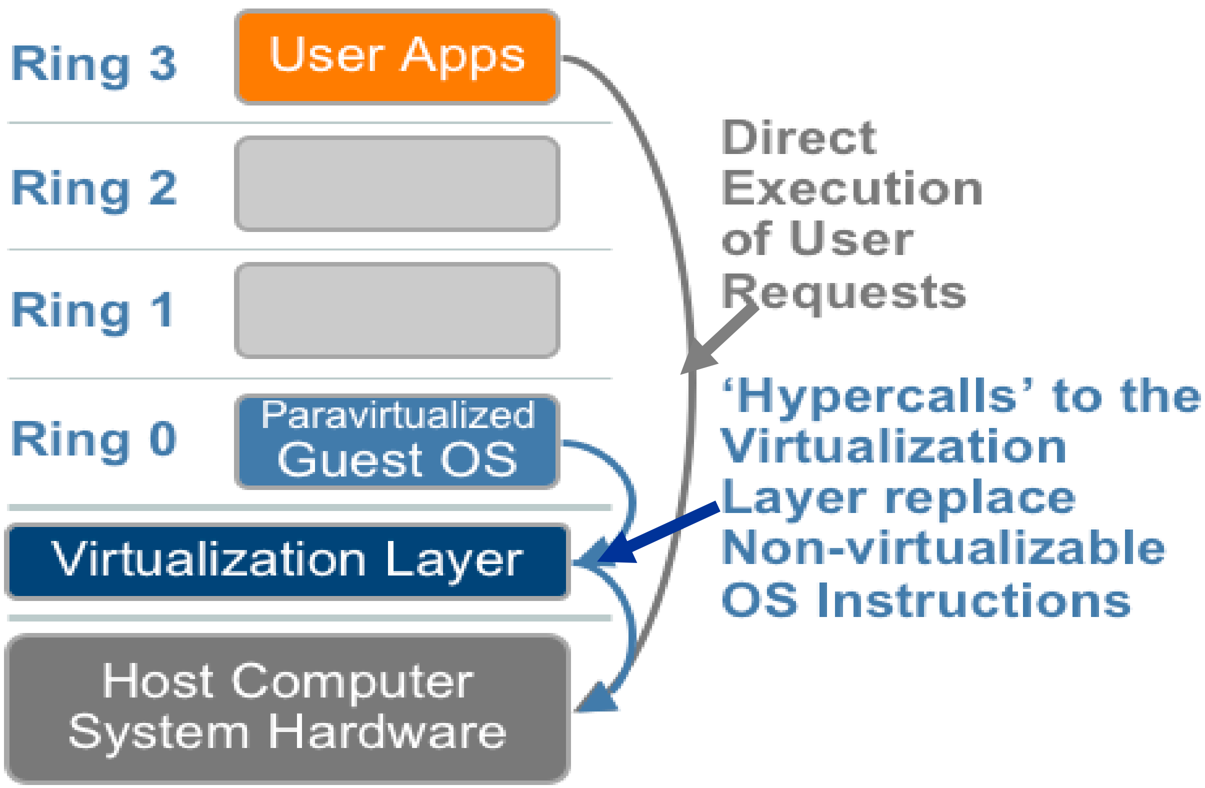
OS Assisted Virtualization or Paravirtualization

- Paravirtualization refers to the communication between the guest OS and the hypervisor to improve performance and efficiency
- It involves modifying the OS kernel to replace non-virtualizable instructions with hyper calls that communicate directly with the virtualization layer hypervisor.

Syscall and Hypercall

- A system call, or syscall, is the mechanism used by an application program to request service from the operating system.
- A hypervisor call, or hypercall, referred to the paravirtualization interface, by which a guest operating system could access hypervisor services.

X86 processor Para-Virtualization Architecture



- Ex. Xen, KVM, and VMware ESX

Figure Courtesy: Understanding Full Virtualization, Paravirtualization, and Hardware Assist, VMware

OS Assisted Virtualization or Paravirtualization

- The hypervisor also provides hypercall interfaces for other critical kernel operations such as memory management, interrupt handling and time keeping
- Paravirtualization is different from full virtualization, where the unmodified OS does not know it is virtualized and sensitive OS calls are trapped using binary translation.
- Paravirtualization cannot support unmodified operating systems (e.g. Windows 2000/XP)

KVM

- KVM (Kernel-Based VM) is a Linux para-virtualization (2.6.20 kernel)
- Memory management and scheduling activities are carried out by the existing Linux kernel. The KVM does the rest
- KVM is a hardware-assisted para-virtualization tool

Para-Virtualization with Compiler Support

- While full virtualization architecture intercepts and emulates privileged and sensitive instructions at runtime, para-virtualization handles these instructions at compile time.
- Ex. Xen assumes such a para-virtualization architecture
- Guest OS running in a guest domain may run at Ring 1 instead of at Ring 0.

Hardware Assisted Virtualization

- Hardware vendors are rapidly accepting the virtualization and developing new features to simplify virtualization techniques.
- Intel Virtualization Technology (VT-x) and AMD's AMD-V are the first wave.
- Above both target *privileged instructions with a new CPU execution mode feature* that allows the VMM to run in a ***new root mode below ring 0***.

The hardware assist approach to X86 virtualization

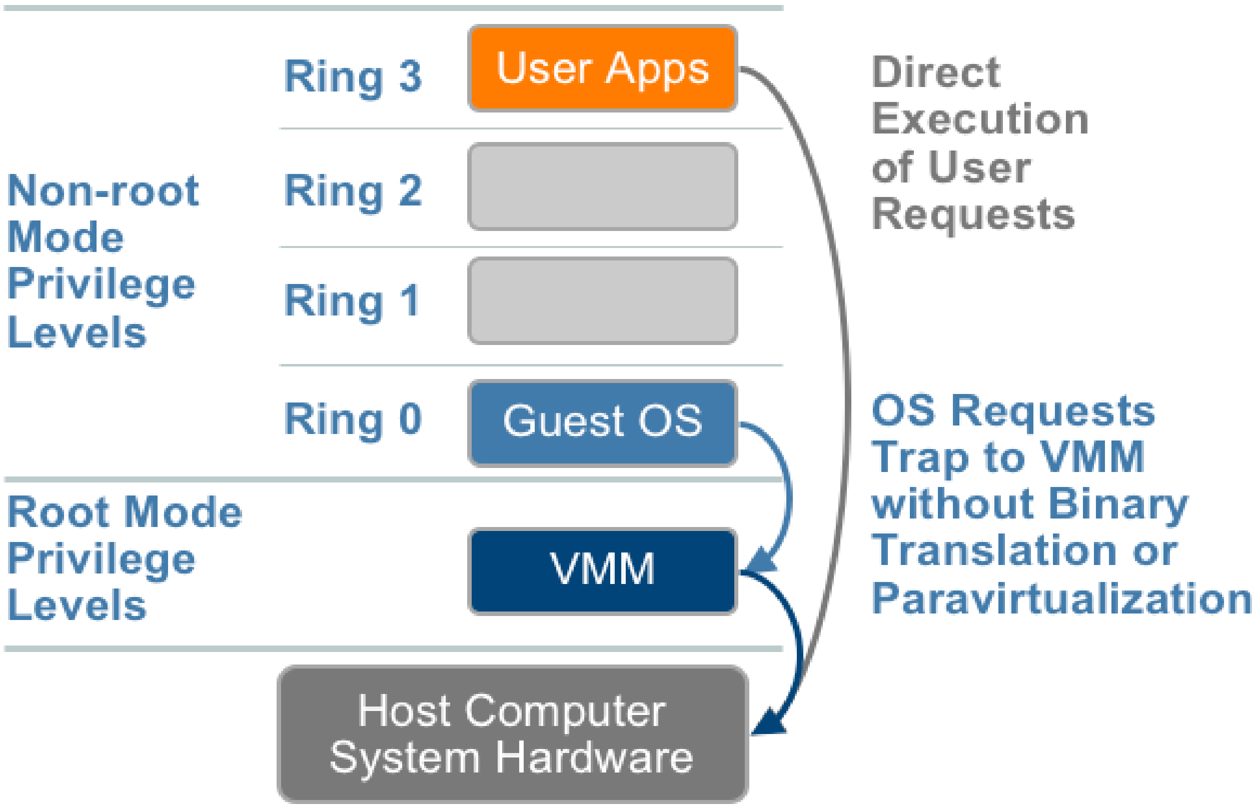


Figure Courtesy: Understanding Full Virtualization, Paravirtualization, and Hardware Assist, VMware

Hardware Assisted Virtualization

- Privileged and sensitive calls are set to automatically trap to the hypervisor, removing the need for either binary translation or paravirtualization
- The guest state is stored in Virtual Machine Control Structures (VT-x) or Virtual Machine Control Blocks (AMD-V).
- First appeared on the IBM System/370 in 1972
  - Ex. Linux KVM, VMware Workstation, VMware Fusion, Microsoft Hyper-V, Microsoft Virtual PC, Xen, Parallels Desktop for Mac, Oracle VM Server for SPARC, VirtualBox and Parallels Workstation.

Intel Hardware-Assisted CPU Virtualization